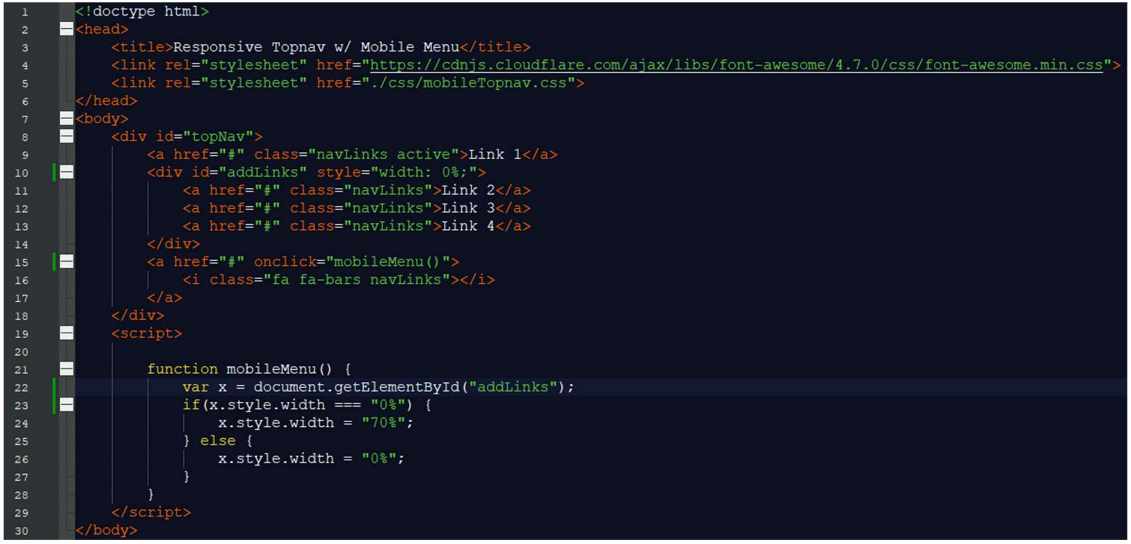
Creating a Flexible Top Navigation (with Mobile Menu)

It's important to note that this tutorial was created utilizing Notepad++ and the Google Chrome web browser. If you're not using both aforementioned programs, your screens will not look exactly the same, and you may need to do some looking around for what you need.

This assignment will show you one of many ways to create a flexible, responsive, top navigation bar. Considering it uses both some elements from the previous assignment and new elements, I will only be explaining the new elements in this walkthrough. This task will also utilize a little bit of JavaScript, the instructional scripts for elements on a website, so I will be providing what you need to make that function, as well as info about why it functions the way it does.

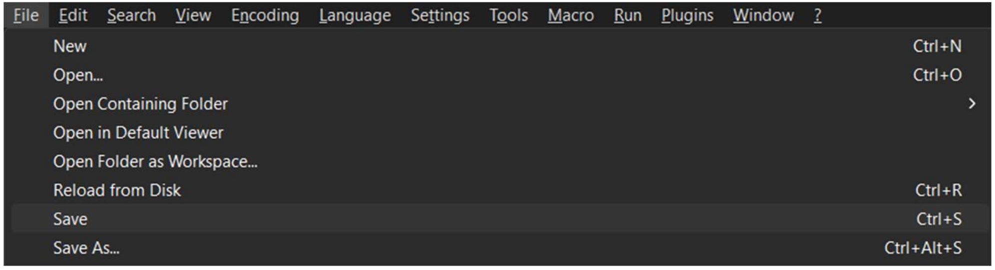
1. First and foremost, we'll start will all the code we need in our HTML document. Remember that HTML is the information and overall structure of the page. It has nothing to do with the visual appeal or more advanced functionality like CSS and JavaScript, respectively.
   To complete this step, make sure you set your new file's language to HTML in the "Language" menu across the top.



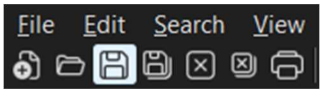
```
<!doctype html>
<head>
    <title>Responsive Topnav w/ Mobile Menu</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="./css/mobileTopnav.css">
</head>
<body>
    <div id="topNav">
        <a href="#" class="navLinks active">Link 1</a>
        <div id="addLinks" style="width: 0%;">
            <a href="#" class="navLinks">Link 2</a>
            <a href="#" class="navLinks">Link 3</a>
            <a href="#" class="navLinks">Link 4</a>
        </div>
        <a href="#" onclick="mobileMenu()">
            <i class="fa fa-bars navLinks"></i>
        </a>
    </div>
    <script>

        function mobileMenu() {
            var x = document.getElementById("addLinks");
            if(x.style.width === "0%") {
                x.style.width = "70%";
            } else {
                x.style.width = "0%";
            }
        }
    </script>
</body>
```

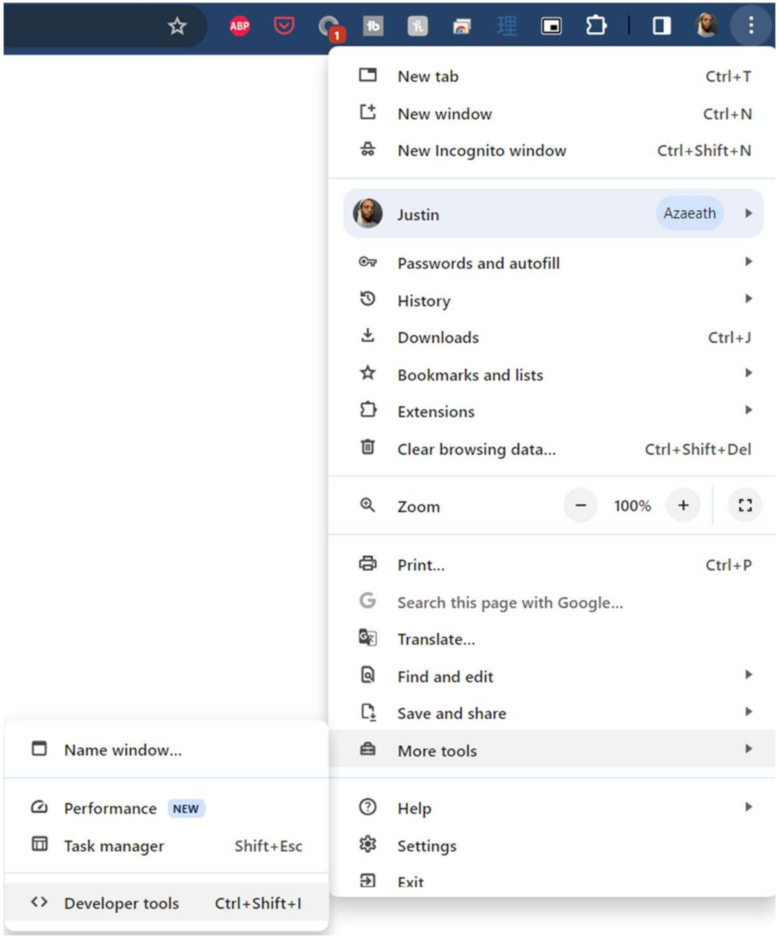
2. The next thing is to save the file somewhere on your computer. Because some people had some problems figuring out where this is, I will list the three ways you can do this:
   a. The first and commonly used method is to go to the file menu in the top left corner of the window, and press save.



   b. The second and quicker method that may even be more widespread is to hold the Control (Ctrl) key on the keyboard and press the letter S to bring up the same save menu.
   c. The third, not always employed method, is to press the save icon that looks like a Floppy Disk in the top left corner.

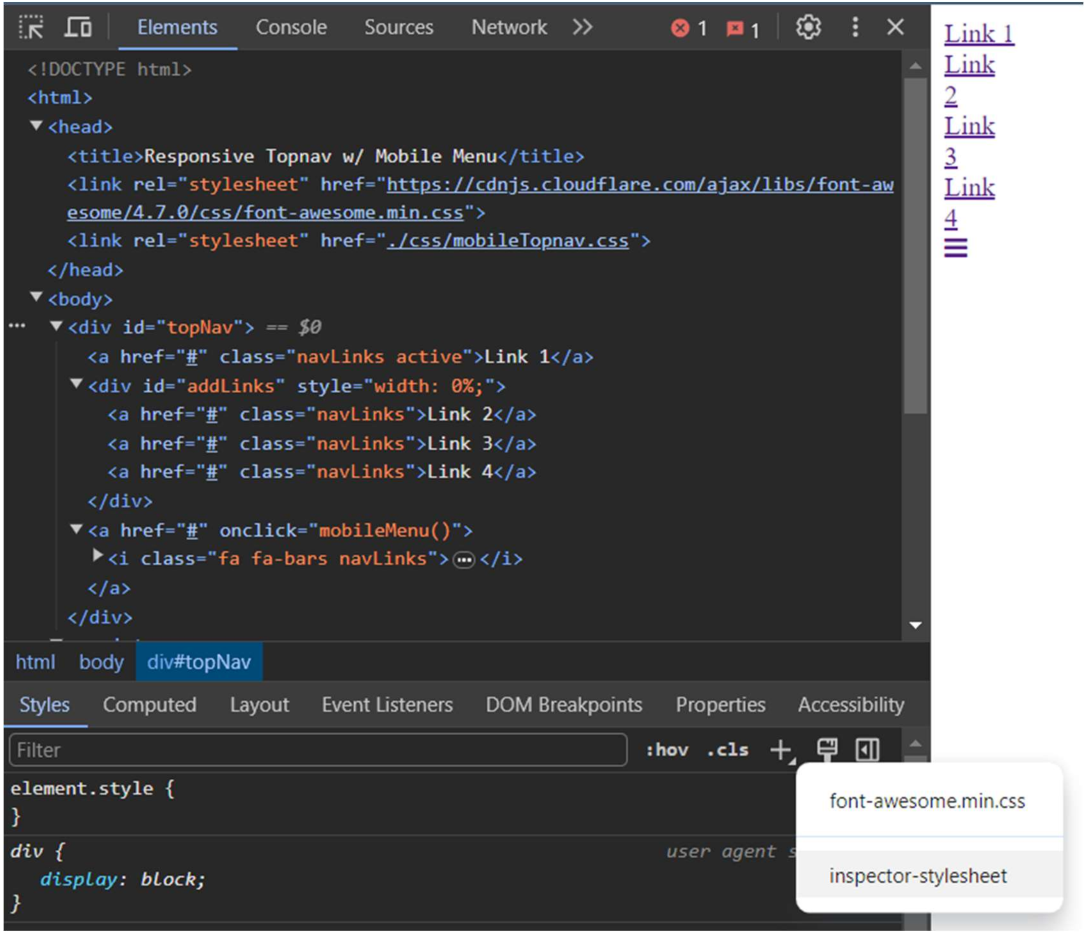


3. When you first open the file in your web browser after saving it, the page should look, more or less, blank, with the exception of the links and menu bar that should now be showing.
4. Just like in the previous assignment, you want to open your developer tools by either holding the Ctrl and Shift keys, simultaneously, and pressing the letter I on the keyboard or clicking the menu in the top right of your browsr, which usually looks like 3 dots, mousing over More Tools, and clicking the Developer Tools option.



Link 1
Link 2
Link 3
Link 4
≡

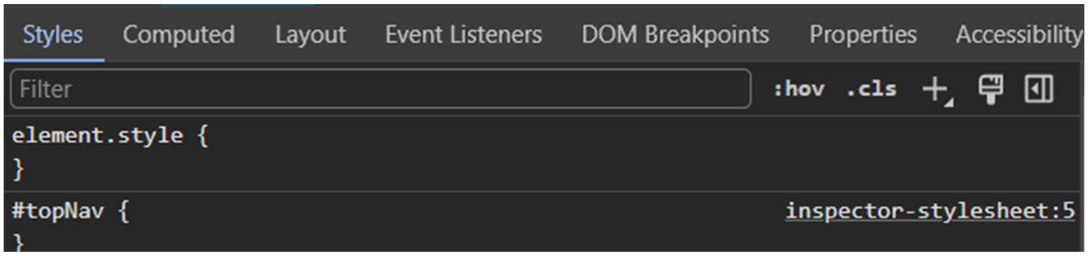
5. Click any element (div, a, etc.) in the Elements tab, then mouse over to the Styles tab down below, pressing the plus icon to create a new Inspector Stylesheet.



```
<!DOCTYPE html>
<html>
▼<head>
   <title>Responsive Topnav w/ Mobile Menu</title>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
   <link rel="stylesheet" href="./css/mobileTopnav.css">
 </head>
▼<body>
  ▼<div id="topNav"> == $0
     <a href="#" class="navLinks active">Link 1</a>
    ▼<div id="addLinks" style="width: 0%;">
       <a href="#" class="navLinks">Link 2</a>
       <a href="#" class="navLinks">Link 3</a>
       <a href="#" class="navLinks">Link 4</a>
     </div>
    ▼<a href="#" onclick="mobileMenu()">
      ▶<i class="fa fa-bars navLinks">⋯</i>
     </a>
   </div>
```

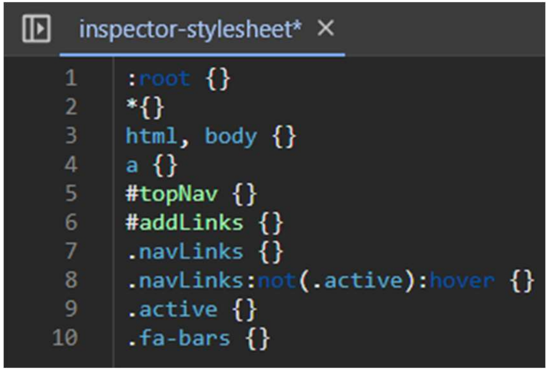
```
element.style {
}
div {
  display: block;
}
```

6. In the styles window, click the resulting link that was created to get to the Inspector Stylesheet in the Sources tab.



```
element.style {
}
#topNav {                                inspector-stylesheet:5
}
```

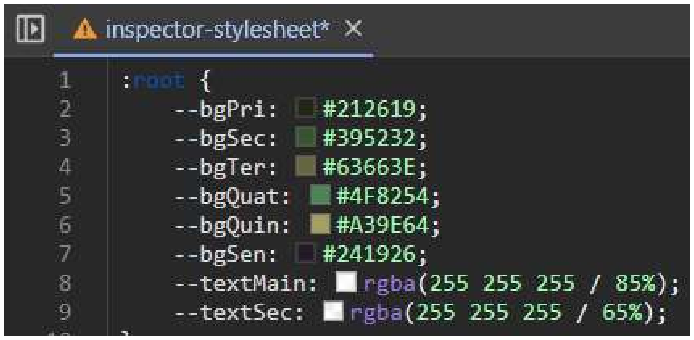
7. Now that we're past basic computing, input the following lines of CSS styling into your Inspector Stylesheet:



```
:root {}
*{}
html, body {}
a {}
#topNav {}
#addLinks {}
.navLinks {}
.navLinks:not(.active):hover {}
.active {}
.fa-bars {}
```

   a. The :root selector allows us to input variables to be referenced later. I primarily use them for colors, but they can be used for other values as well.
   b. The * (asterisk, a.k.a. star) selector enables us to edit all elements on the page in one go.
   c. The html selector allows us to make edits to the document overall, while the body selector allows us to make edits to the default container of the elements being displayed on the page.

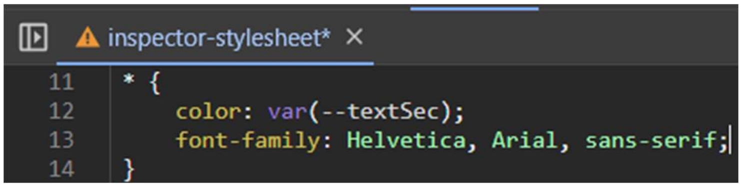
8. Now that we have all the selectors in place, the next step is to start filling in the names and values of all the variables we'll want to use in our color scheme.
   a. Pay careful attention to the spelling and punctuation when typing these things if you haven't been already. That is the only way to mess this up.



```
:root {
    --bgPri: #212619;
    --bgSec: #395232;
    --bgTer: #63663E;
    --bgQuat: #4F8254;
    --bgQuin: #A39E64;
    --bgSen: #241926;
    --textMain: rgba(255 255 255 / 85%);
    --textSec: rgba(255 255 255 / 65%);
```

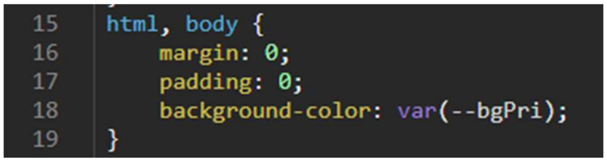
   b. All the variables must have the two hypens (-) in front of their names. However, you can name them all whatever you'd like. The main thing to keep in mind is that you spell them the same way you name them when you start to reference them later on. That is to say that if you name them "bgPri," like it says in the picture, you must spell it the same way, including the capitalization, when you reference the variable later.
   c. The names of the variables you see here are abbreviations of the following words:
      i. Background – bg
      ii. Primary – Pri
      iii. Secondary – Sec
      iv. Tertiary – Ter
      v. Quaternary – Quat
      vi. Quinary – Quin
      vii. Senary – Sen

9. Next, we'll change the color and font of all the text on the page. This will also affect our anchor tags (site links), so we'll be making minor changes to those separately during a later step. For now, add these two lines:



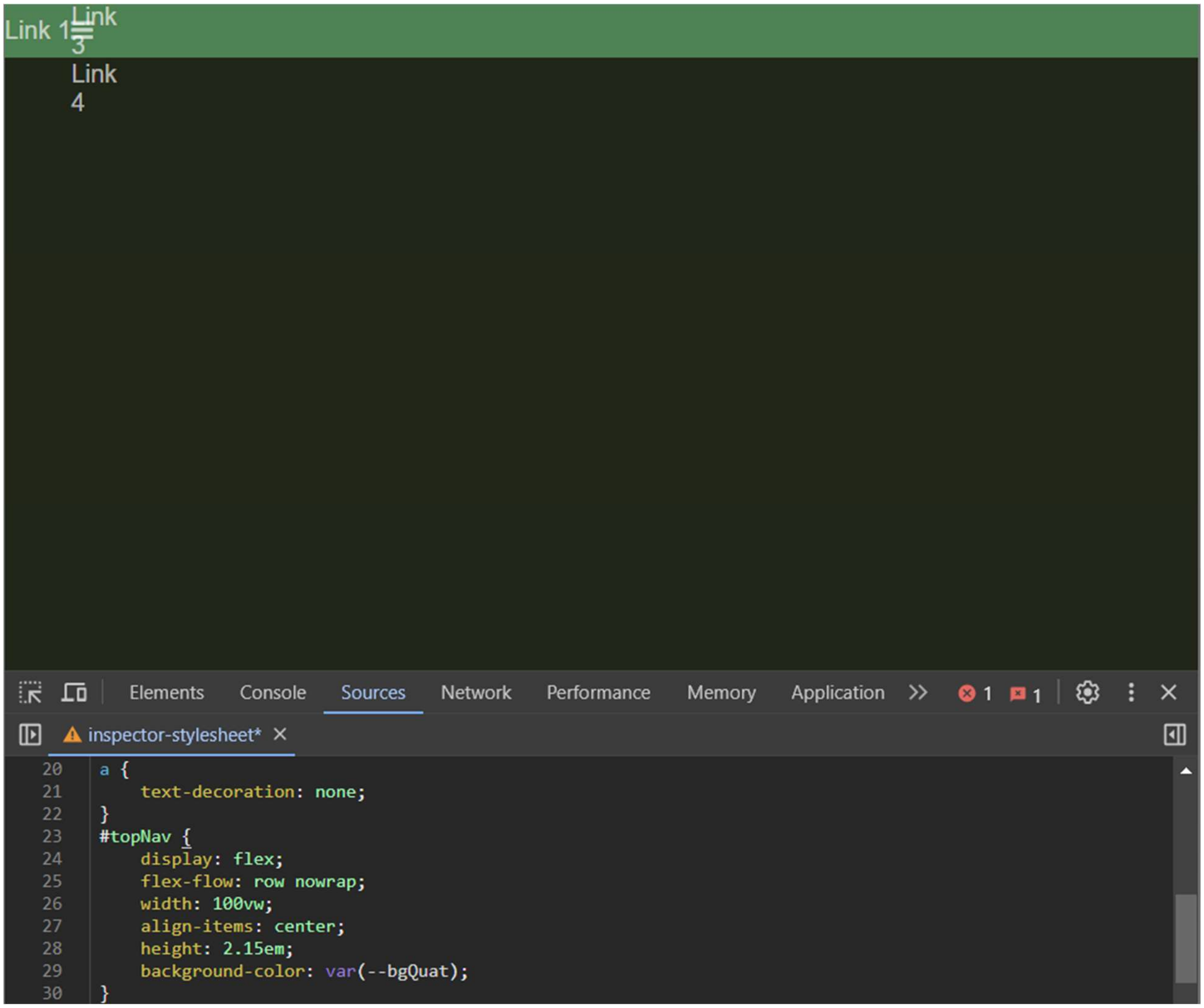
```
* {
    color: var(--textSec);
    font-family: Helvetica, Arial, sans-serif;
}
```

10. Because the type of navigation bar we’re making is meant to look like tabs at the end, we’re going to go ahead and apply a background color to the page, as well as making it so the elements we add later can take up as much screen space as possible, meaning they’ll be able to touch the edges of the web page unless we move them away manually.



```
html, body {
    margin: 0;
    padding: 0;
    background-color: var(--bgPri);
}
```

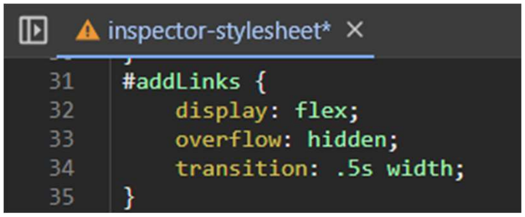
11. Add the additional lines in the image below:



```
a {
    text-decoration: none;
}
#topNav {
    display: flex;
    flex-flow: row nowrap;
    width: 100vw;
    align-items: center;
    height: 2.15em;
    background-color: var(--bgQuat);
}
```

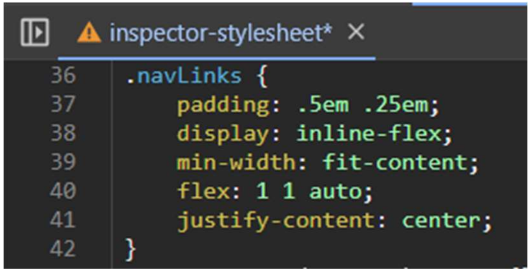
By the time you finish, your web page should be looking something like the above. If not, figure it out. Somewhere along the way you missed a punctuation (or more) or you mispelled something, if not multiple things.

12. The next few lines we’re adding will clean up the appearance and organization of the links we’ve added to the bar, as well as hide them from view until we press our menu button.



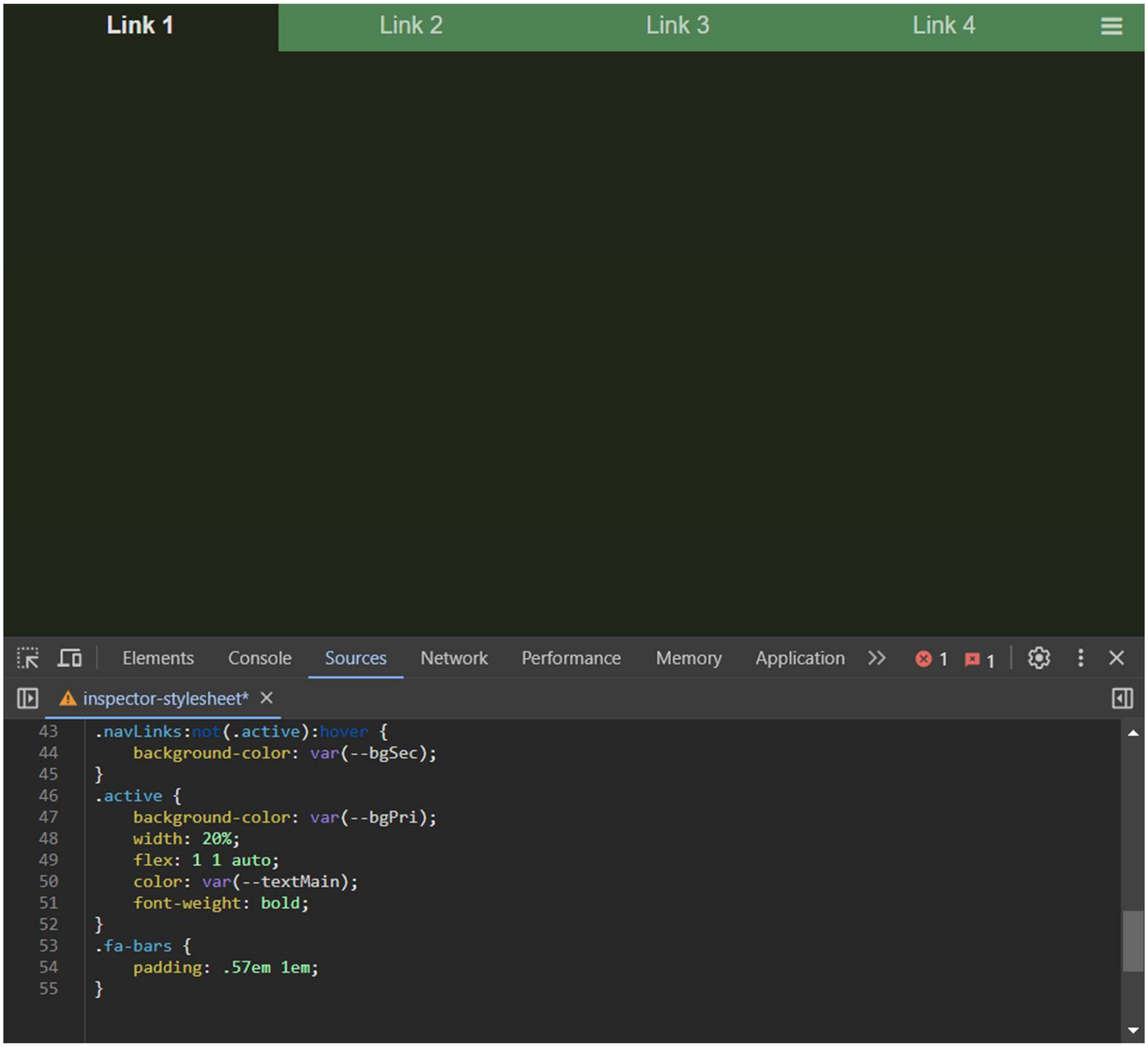
```
#addLinks {
    display: flex;
    overflow: hidden;
    transition: .5s width;
}
```

13. Additionally, we’ll be making it so that the links on the bar will resize themselves to fit the space they have available on screen. This method is the part that makes them “responsive.” The part that does this, particularly, is the flex property that is shorthand for the flex-grow, flex-shrink, and flex-basis properties all in one.



```
.navLinks {
    padding: .5em .25em;
    display: inline-flex;
    min-width: fit-content;
    flex: 1 1 auto;
    justify-content: center;
}
```

14. Because we’re in the last stretch of the assignment, the next section will be a three-in-one combo screenshot:

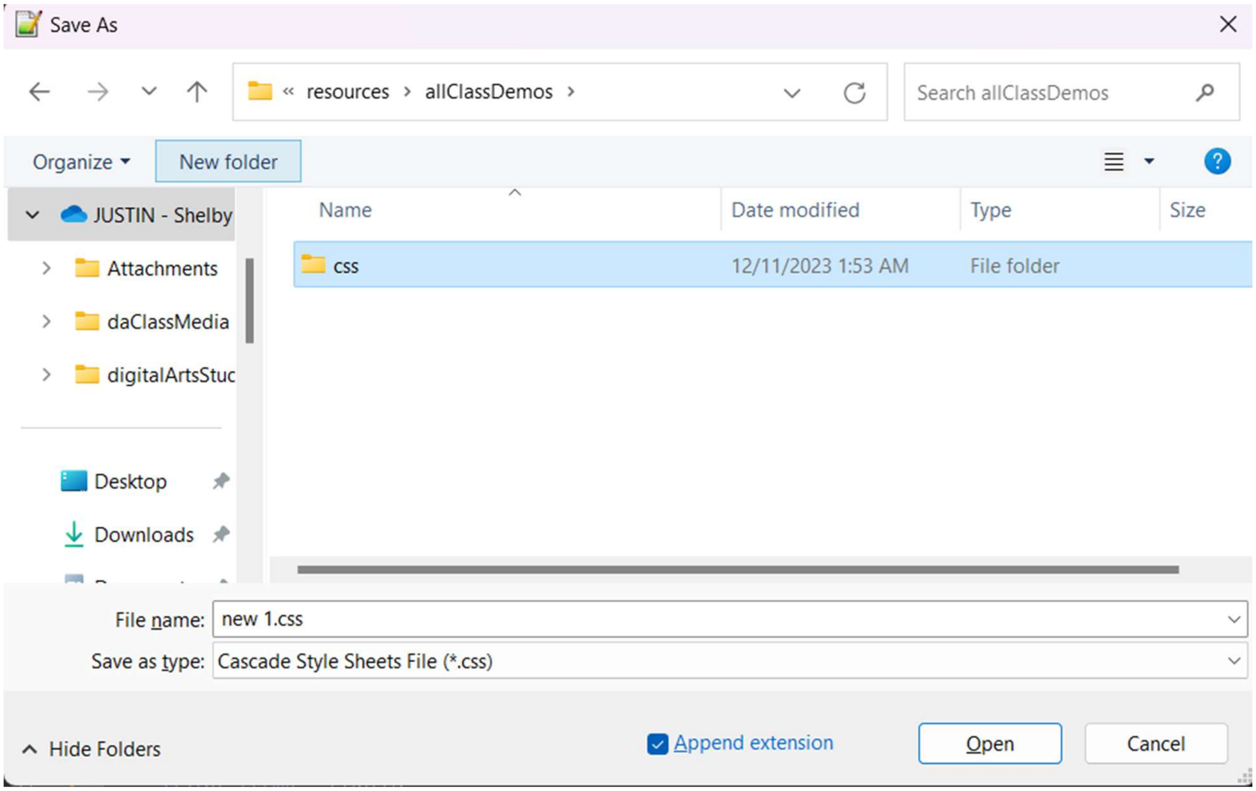


```
43  .navLinks:not(.active):hover {
44      background-color: var(--bgSec);
45  }
46  .active {
47      background-color: var(--bgPri);
48      width: 20%;
49      flex: 1 1 auto;
50      color: var(--textMain);
51      font-weight: bold;
52  }
53  .fa-bars {
54      padding: .57em 1em;
55  }
```

a. Lines 43-45 basically state that anything you hover over that is not the active link for the current page will darken to show that you can interact with it.
b. Lines 46-52 style the active link to have a background color that blends with the overall page, allowing it to look like a tab, sets its default width to 20%, allows for growing and shrinking like the other links on the bar, and makes the text a bit brighter and bolder.
c. Lines 53-55 are for the menu button. The padding added to it just makes the clickable area around the button’s three bars larger and easier to click.

15. At this point, in your inspector stylesheet, hold the Control key and press A to select all the text in the stylesheet. You can then right-click or just hold Control and press C to copy it all. We’ll be pasting it back into Notepad++ as its own new file.
a. You’ll want to set the language to CSS on this new file.

16. When saving the new CSS file, make sure you make a folder that's in the same root folder as your HTML file. You then want to name that folder css.



Name	Date modified	Type	Size
css	12/11/2023 1:53 AM	File folder	

File name: new 1.css

Save as type: Cascade Style Sheets File (*.css)

☑ Append extension

17. Lastly, go ahead and name the file "mobileTopnav," barring the quotations and that comma, and save it to the new "css" folder you just made. Together the two files should be linked and working in tandem when you reload the page in your web browser. You should now be done with your mobile top navigation. If you run into anything regarding the appearance or functionality, go back through your code and make sure you don't have any spelling or punctuation errors. Those are the only two ways you can mess this up.